Technical Note

Wang, Y.; Mi, X.; Rosa, G. J. M.; Chen, Z.; Lin, P.; Wang, S.

Published in:
Journal of Animal Science

DOI:
10.1093/jas/sky071

Publication date:
2018

Document Version
Peer reviewed version

Technical Note: An R package for fitting sparse neural networks with application in animal breeding[1]

Sparse neural networks in animal breeding

Y. Wang,* X. Mi,* G. J. M. Rosa, #[2] Z. Chen, † P. Lin,‡ S. Wang,* and Z. Bao*[2]

*****Ministry of Education Key Laboratory of Marine Genetics and Breeding, Ocean University of China, Qingdao, China, 266003;

#Department of Animal Sciences, University of Wisconsin, Madison, 53706;

†College of Life Science, University of Dundee, UK, DD1 4HN;

‡Division of Mathematics, University of Dundee, UK, DD1 4HN.

[1]This study is supported by the National Natural Science Foundation of China (No. 31772844 , 31302182).

[2]Corresponding author: grosa@wisc.edu or zmbao@ouc.edu.cn

This is a pre-copyedited, author-produced version of an article accepted for publication in *Journal of Animal Science* following peer review. The version of record Wang, Y., et al. (2018) 'Technical Note: An R package for Fitting Sparse Neural Networks with Application in Animal Breeding', *Journal of Animal Science*, is available online at: https://dx.doi.org/10.1093/jas/sky071

ABSTRACT: Neural networks (NN) have emerged as a new tool for genomic selection (GS) in animal breeding. However, the properties of NN used in GS for the prediction of phenotypical outcomes are not well characterized due to the problem of over-parameterization of NN and difficulties in using whole-genome marker sets as high-dimensional NN input. In this note, we have developed an R package called snnR that could find an optimal sparse structure of a NN by minimizing the square error subject to a penalty on the L_1-norm of the parameters (weights and biases), therefore solve the problem of over-parameterization in NN. We have also tested some models fitted in the snnR package to demonstrate their feasibility and effectiveness to be used in several cases as examples. In comparison of snnR to the R package brnn (the Bayesian regularized single layer neural networks), though both using the entries of a genotype matrix or a genomic relationship matrix as inputs, snnR has greatly improved the computational efficiency and the prediction ability for the GS in animal breeding because snnR implements a sparse neural network with many hidden layers.

Key words: animal breeding, sparse neural networks, genomic selection, dominance and additive effects, genetic markers

INTRODUCTION

Genomic selection (**GS**) is an essential process in modern animal breeding for obtaining desired phenotypes or traits because large data about both genomic information and phenotypes are accumulated and available for analysis (Meuwissen et al., 2001). However, the relationships between genomic information or genes and phenotypes are not straightforward quantitatively linked. Therefore, some statistical models have been applied to predict such relationships, especially for these complex traits (Gianola et al., 2008). Traditional models have defects because they typically ignored those genes that have complicated interactions with each other or higher order non-linearity in determining their corresponding phenotypes. To take possible non-linearity into account in prediction, neural networks (**NN**) have emerged as a new tool of GS for marker-based genomic predictions of complex traits in animal breeding (Gianola et al., 2011; Okut et al., 2013; Pérez-Rodríguez et al., 2013; Ehret et al., 2015).

Moreover, due to the problem of over-parameterization when using whole-genome marker sets as high-dimensional NN input, it is not easy to construct an efficient NN for the prediction of future outcomes of GS (Ehret et al., 2015). Many dimension reduction methods, such as genome-derived relationships among individuals (VanRaden, 2008) or singular value decomposition of whole-genome marker sets (Pérez-Rodríguez et al., 2013) as NN inputs, have been used to enhance prediction performance thanks to the decreasing computational cost (Ehret et al.,

2015). It is believed that detecting an optimal sparse structure of a NN still remains as an effective way to reduce the amount of computational costs (Anders and Korn, 1999; Gripon and Berrou, 2011; Thomaidis et al., 2011). The sparse deep NN with high-dimensional inputs has been shown to be extremely powerful in connection with performing classification tasks and was widely used in the area of machine learning (Gripon and Berrou, 2011; Scardapane et al,2017). Examples for fitting deep neural networks are the *deepnet* R package from CRAN, the *DNN* function in Tensorflow Python package and the *Deep Learning* with Matlab. However, very few existing software packages of the sparse NN were available in either the public or commercial domains, not to mention the application in GS.

In this note, we have developed an R package called **snnR** that finds an optimal sparse structure of a NN by minimizing the square error subject to a penalty on the L_1-norm of the parameters (weights and biases), which was introduced in the context of linear regression by Tibshirani (1996) known as the lasso estimator, to solve the over-parameterization in NN with a genomic relationship matrix or a SNP marker matrix as input for improving the predictive performance in the GS of animal breeding. To solve the problems of non-differentiability of the L_1-norm penalty, we use a subgradient-based quasi-Newton method (Bertsekas et al., 2003). It is based on choosing a subgradient with minimum norm as a steepest descent direction and taking a step resembling Newton iteration in this direction with a Hessian approximation (Nocedal, 1980). An active-set method is adopted to set some parameters to exactly zero (Krishnan et al., 2007). The application of snnR is extended, such that additive

and dominance effects can be fitted for genome-enabled selection. snnR also implements a feed-forward neural network with many hidden layers to improve the modeling power to predict complex traits more than when single layer architecture is used in NN. The package is available at the CRAN site (Comprehensive R Archive Network, http://cran.r-project.org/mirrors.html).

METHODS

Linear, Nonlinear Models, and Neural Networks

Meuwissen et al. (2001) introduced the use of linear regression models in genome-enabled predictions. The basic linear regression model for additive effects is

$$y_i = \mu + \sum_{j=1}^{p} x_{ij}\,\beta_j + \epsilon_i, \qquad [1]$$

where y_i is a target trait measured on individual i; μ is an intercept; β_j is the allele substitution effect of marker j; x_{ij} is the jth marker genotype observed in individual i; and $\epsilon_i \sim NIID(0,\ \sigma_e^2)$, where σ_e^2 is the residual variance. A more general regression model suitable for capturing nonlinear patterns can be written as

$$y_i = \mu + g(\boldsymbol{x_i}) + \epsilon_i, \qquad [2]$$

where $g(.)$ is a function that maps from the input space (p-dimensional) to the real line, $\boldsymbol{x_i}$ is the set of genotypic codes observed on i and ϵ_i is as in Eq. [1]. It is well known that any nonlinear function can be exactly represented by a NN (Kurkova,1992).

The Single Hidden Layer Feed Forward Neural Networks (**SLNN**) for GS is introduced by Gianola et al. (2011) :

$$y_i = \mu + \sum_{k=1}^{S} W_k \, g_k \left(b_k + \sum_{j=1}^{p} x_{ij} \, \beta_j^{[k]} \right) + \epsilon_i. \qquad [3]$$

In terms of genome-enabled prediction using [3], in the hidden layer, the genomic covariates x_{ij} (for $j = 1, \cdots, p$) of an individual i (for $i = 1, \cdots, n$) are linearly combined with a vector of input weights $\beta_j^{[k]}$ that are specified in the training phase, plus an intercept (in NN's terminology also called "bias") b_k with $k = 1, \cdots, S$ denoting a neuron. The resulting linear score is then transformed using an activation function $g_k(.)$ to produce the output of the single hidden neuron. To model non-linear relationship between phenotype and input, the tangent hyperbolic function ($\tanh(x) = \frac{2}{[1+exp(-2x)]} - 1$) can be used in the hidden neurons. In the output layer, the S genotype-derived basis functions, resulting from the hidden layer, are also linearly combined by using the $W_1, W_2, \cdots, W_S$ weights.

The NN model given by Eq. [3] can be fitted by using as predictors: i) the incidence matrix for additive effects of genetic markers $\mathbf{X} = \{x_{ij}\} \in \{-1,0,1\}$ with dimensions $n \times p$, or ii) a genomic relationship matrix (e.g., **G**), as pointed out by Gianola et al. (2011). For example, a genomic relationship matrix suggested by VanRaden(2008) as :

$$\mathbf{G} = \frac{\mathbf{x}\mathbf{x}'}{2\sum_{j=1}^{p} p_j(1-p_j)}, \qquad [4]$$

where $\mathbf{X}$ is the incidence matrix for additive effects and p_j is the minor allele frequency for SNP j, $j = 1, \cdots, p$.

A NN model that includes additive and dominance effects jointly is presented by Pérez-Rodríguez et al. (2013) as follows:

$$y_i = \mu + \sum_{k=1}^{S_a} W_k^a \, g_k^a \left(b_k^a + \sum_{j=1}^{p} x_{ij} \, \beta_j^{a[k]} \right) + \sum_{k=1}^{S_d} W_k^d \, g_k^d \left(b_k^d + \sum_{j=1}^{p} z_{ij} \, \beta_j^{d[k]} \right) + \epsilon_i, \quad [5]$$

where $\mathbf{Z} = \{z_{ij}\} \in \{0,1\}$ is the incidence matrix for dominance effects of dimension n $\times p$, if markers are used, $z_{ij} = 1$ if SNP j for individual i is heterozygous, and $z_{ij} = 0$ otherwise. Further, S_a and S_d are the numbers of neurons for the additive and dominance components in the hidden layers, respectively.

The NN model given by Eq. [5] can be fitted by using as predictors: i) the incidence matrix for additive effects of genetic markers $\mathbf{X}$ and the incidence matrix for dominance effects of markers $\mathbf{Z}$, or ii) the genomic additive relationship matrix $\mathbf{G}$ and the genomic dominance relationship matrix $\mathbf{D}$ using the same arguments of VanRaden (2008) as:

$$\mathbf{D} = \frac{\mathbf{Z}\mathbf{Z}'}{2\sum_{j=1}^{p} p_j \, q_j (1 - 2 p_j \, q_j)}, \quad [6]$$

where $q_j = 1 - p_j$.

Sparse Neural Networks

We obtain an estimate of sparse structure of model [3] by minimizing the negative logarithm of likelihood of the data with sparsity enforcing L_1-norm penalty on parameters $\left\{W_k, b_k, \beta_j^{[k]}\right\}$ $(k = 1, \cdots, S\ ;\ \ j = 1, \cdots, p)$ as follows:

$$\min_{W_k, b_k, \beta_j^{[k]}} \tilde{F}\left(W_k, b_k, \beta_j^{[k]}\right)$$

$$\triangleq \hat{\mathcal{L}}\left(W_k, b_k, \beta_j^{[k]}\right) + \left(\sum_{k=1}^{S}\sum_{j=1}^{p}\lambda_{k,j}|\beta_j^{[k]}| + \sum_{k=1}^{S}\lambda_k|b_k| + \sum_{k=1}^{S}\lambda_k|W_k|\right), \quad [7]$$

where the approximate square error

$\hat{\mathcal{L}}\left(W_k, b_k, \beta_j^{[k]}\right) = \sum_{i=1}^{n}\left(\sum_{k=1}^{S} W_k\, g_k\left(b_k + \sum_{j=1}^{p} x_{ij}\,\beta_j^{[k]}\right) - y_i\right)^2$, $\lambda_{k,j}(\lambda_{k,j} > 0)$ and $\lambda_k(\lambda_k > 0)$ are Lagrange multipliers that determine the amount of sparsity in $\beta_j^{[k]}$, W_k and b_k. Note that μ is not included in $\hat{\mathcal{L}}\left(W_k, b_k, \beta_j^{[k]}\right)$, as this parameters can be easily eliminated, for example, simply by centering the response vector.

The gradients of $\hat{\mathcal{L}}\left(W_k, b_k, \beta_j^{[k]}\right)$ with respect to $W_k, b_k, \beta_j^{[k]}$ are given by

$$\nabla_{W_k}\hat{\mathcal{L}}\left(W_k, b_k, \beta_j^{[k]}\right) =$$

$$2\sum_{i=1}^{n}\left(\sum_{k=1}^{S} W_k\, g_k\left(b_k + \sum_{j=1}^{p} x_{ij}\,\beta_j^{[k]}\right) - y_i\right) g_k\left(b_k + \sum_{j=1}^{p} x_{ij}\,\beta_j^{[k]}\right), \quad [8]$$

$$\nabla_{b_k}\hat{\mathcal{L}}\left(W_k, b_k, \beta_j^{[k]}\right) =$$

$$2\sum_{i=1}^{n}\left(\sum_{k=1}^{S} W_k\, g_k\left(b_k + \sum_{j=1}^{p} x_{ij}\,\beta_j^{[k]}\right) - y_i\right) W_k \dot{g}_k\left(b_k + \sum_{j=1}^{p} x_{ij}\,\beta_j^{[k]}\right), \quad [9]$$

where $\dot{g}$ denotes the derivative of g; and

$$\nabla_{\beta_j^{[k]}} \hat{\mathcal{L}}\left(W_k, b_k, \beta_j^{[k]}\right) =$$

$$2\sum_{i=1}^{n}\left(\sum_{k=1}^{S} W_k\, g_k\left(b_k + \sum_{j=1}^{p} x_{ij}\,\beta_j^{[k]}\right) - y_i\right) W_k \dot{g}_k\left(b_k + \sum_{j=1}^{p} x_{ij}\,\beta_j^{[k]}\right) x_{ij}. \quad [10]$$

Let $\boldsymbol{\mathcal{X}} = \left(W_1, \cdots, W_S, b_1, \cdots, b_S, \beta_1^{[1]}, \cdots, \beta_S^{[1]}, \cdots, \beta_1^{[p]}, \cdots, \beta_S^{[p]}\right)' \in \boldsymbol{\mathcal{R}}^{S+S+S*p}$ in model [3] be the vector of weights, biases. Thus, the [7] can be written as

$$\min_{\boldsymbol{\mathcal{X}}} \tilde{F}(\boldsymbol{\mathcal{X}}) \triangleq \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) + \lambda_r \sum_{r=1}^{S+S+S*p} |\mathcal{X}_r|, \quad [11]$$

where $\boldsymbol{\mathcal{X}} = \left(W_1, \cdots, W_S, b_1, \cdots, b_S, \beta_1^{[1]}, \cdots, \beta_S^{[1]}, \cdots, \beta_1^{[p]}, \cdots, \beta_S^{[p]}\right)' \in \boldsymbol{\mathcal{R}}^{S+S+S*p}$.

Subgradient-based Method to Solve the Sparse Neural Networks

In [11], $\hat{\mathcal{L}}(\boldsymbol{\mathcal{X}})$ of $\tilde{F}(\boldsymbol{\mathcal{X}})$ is not convex and differentiable with respect to $\mathcal{X}_r$. Solving [11] is complicated by the non-differentiability of $|\mathcal{X}_r|$ at $\mathcal{X}_r = 0$. Subgradient methods are among the most popular method for non-differentiable optimization (Bertsekas et al., 2003). We use the subgradient with minimum norm (Bertsekas et al., 2003) of $\tilde{F}(\boldsymbol{\mathcal{X}})$ in [11] as the steepest descent direction and take a step resembling a Newton iteration in this direction with a Hessian approximation(Gill et al., 1984) to solve the above problem.

The subgradient of $\tilde{F}(\boldsymbol{\mathcal{X}})$ in [11] with respect to a variable $\mathcal{X}_r$ is given by

$$\partial_r \tilde{F}(\boldsymbol{\mathcal{X}}) = \nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) + \lambda_r sgn(\mathcal{X}_r), \quad [12]$$

where $\partial_r \tilde{F}(\boldsymbol{\mathcal{X}}) = \partial_{\mathcal{X}_r} \tilde{F}(\boldsymbol{\mathcal{X}})$, $\nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) = \nabla_{\mathcal{X}_r} \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}})$, and the set-valued function $sgn(\mathcal{X}_r)$ (Bertsekas et al., 2003) is given by

$$sgn(\mathcal{X}_r) \triangleq \begin{cases} sign(\mathcal{X}_r), & \mathcal{X}_r \neq 0 \\ [-1,1], & \mathcal{X}_r = 0 \end{cases}. \quad [13]$$

Because the subgradient of $\tilde{F}(\boldsymbol{\mathcal{X}})$ is separable in the variables, the problems of computing the minimum-norm element of the subgradient is also separable. Hence, we can solve the minimum-norm problem coordinate-wise to yield that the element of the minimum-norm subgradient with respect to a variable $\mathcal{X}_r$ is:

$$\tilde{\nabla}_r \tilde{F}(\boldsymbol{\mathcal{X}}) \triangleq \begin{cases} \nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}), & \boldsymbol{\lambda} = 0 \\ \nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) + \lambda_r sign(\mathcal{X}_r), & \boldsymbol{\lambda} > 0,\ |\mathcal{X}_r| > 0 \\ \nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) + \lambda_r, & \boldsymbol{\lambda} > 0,\ \mathcal{X}_r = 0, \nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) < -\lambda_r \\ \nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) - \lambda_r, & \boldsymbol{\lambda} > 0,\ \mathcal{X}_r = 0, \nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}}) > \lambda_r \\ 0, & \boldsymbol{\lambda} > 0,\ \mathcal{X}_r = 0, |\nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}})| \leq \lambda_r \end{cases} \quad [14]$$

where $\tilde{\nabla}_r \tilde{F}(\boldsymbol{\mathcal{X}}) = \tilde{\nabla}_{\mathcal{X}_r} \tilde{F}(\boldsymbol{\mathcal{X}})$ and $\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_{N_p}\}$ (N_p is the number of parameters $\boldsymbol{\mathcal{X}}$).In the last case of [14], the element of the minimum-norm subgradient is zero, because we can set $sign(\mathcal{X}_r)$ to $-\nabla_r \hat{\mathcal{L}}(\boldsymbol{\mathcal{X}})/\lambda_r$ to achieve a norm of zero.

A search direction to solve the sparse neural networks is as follows

$$\boldsymbol{\hbar}[\kappa] \triangleq -\boldsymbol{\mathcal{H}}^{-1}[\kappa] \tilde{\nabla} \tilde{F}(\boldsymbol{\mathcal{X}}[\kappa]), \quad [15]$$

In dealing with practical problems, quasi-Newton methods (Dennis et al., 1997) allow us to replace the Hessian matrix $\boldsymbol{\mathcal{H}}[\kappa]$ by an approximation $\boldsymbol{\mathcal{B}}[\kappa]$. So, the search direction is changed as follows

$$\boldsymbol{\hbar}[\kappa] \triangleq -\boldsymbol{\mathcal{B}}^{-1}[\kappa] \tilde{\nabla} \tilde{F}(\boldsymbol{\mathcal{X}}[\kappa]), \quad [16]$$

Details on the calculation of the search direction are given in Supplementary A.

Active-set Method to Set Some Parameters to Zero

To set variables $\boldsymbol{\mathcal{X}}[\kappa]$ to exactly zero, we use one of the active-set methods which are widely used for solving L_1-norm regulation problems (Krishnan et al., 2007). The variables are divided into two sets: the working set $\boldsymbol{\mathcal{W}}$ containing the sufficiently non-zero variables, and the active set $\boldsymbol{\mathcal{A}}$ containing the sufficiently zero-values variables.

The working set is defined as follows:

$$\boldsymbol{\mathcal{W}} \triangleq \{r \ |\mathcal{X}_r| > \mathcal{E}\}. \qquad [17]$$

On each iteration, we take a projection of the Newton step along the working set using the Hessian matrix approximation $\boldsymbol{\mathcal{B}}^{-1}[\kappa]$ and a projected minimum norm subgradient for the active-set variables:

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{W}}}[\kappa+1] \leftarrow \mathcal{P}_{\mathcal{O}}\left(\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{W}}}[\kappa] - \iota\boldsymbol{\mathcal{B}}_{\boldsymbol{\mathcal{W}}}^{-1}[\kappa]\tilde{\nabla}_{\boldsymbol{\mathcal{W}}}\tilde{F}(\boldsymbol{\mathcal{X}}[\kappa])\right),$$

$$\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{A}}}[\kappa+1] \leftarrow \mathcal{P}_{\mathcal{O}}\left(\boldsymbol{\mathcal{X}}_{\boldsymbol{\mathcal{A}}}[\kappa] - \iota\boldsymbol{\mathcal{D}}[\kappa]\tilde{\nabla}_{\boldsymbol{\mathcal{A}}}\tilde{F}(\boldsymbol{\mathcal{X}}[\kappa])\right), \qquad [18]$$

where we use $\tilde{\nabla}_{\boldsymbol{\mathcal{A}}}\tilde{F}(\boldsymbol{\mathcal{X}}[\kappa])$ to denote the sub-vector of $\tilde{\nabla}\tilde{F}(\boldsymbol{\mathcal{X}}[\kappa])$ corresponding to elements of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}_{\boldsymbol{\mathcal{W}}}^{-1}[\kappa]$ to denote the sub-matrix of $\boldsymbol{\mathcal{B}}^{-1}[\kappa]$ with all rows and columns of $\boldsymbol{\mathcal{W}}$; $\mathcal{P}_{\mathcal{O}}$ is an orthant projection, which is effective at sparsifying the parameter vector and ensures that the line search does not cross points of non-differentiability , and is defined:

$$\mathcal{P}_{\mathcal{O}}(\boldsymbol{\mathcal{X}} + \boldsymbol{\hbar}) \triangleq \begin{cases} 0 & if \ \ \mathcal{X}_r(\mathcal{X}_r + \hbar_r) < 0 \\ \mathcal{X}_r + \hbar_r & otherwise \end{cases}, \qquad [19]$$

and $\boldsymbol{\mathcal{D}}[\kappa]$ is the diagonal scaling matrix and set to $\sigma^{-1}[\kappa]\mathbf{I}$ with the Barzilai-Borwein scaling $\sigma^{-1}[\kappa]\left(\sigma[\kappa] \triangleq \frac{\mathcal{J}'[\kappa]\mathcal{J}[\kappa]}{\mathcal{J}'[\kappa]\mathcal{S}[\kappa]}\right)$ (Fletcher,1980).

We outline an algorithm that can be used as a basis for implementing a subgradient-based quasi-Newton method for solving the problem [7] (Supplementary B).

EXAMPLES

Jersey Dataset

The first data set, described in detail by Gianola et al. (2011) and made publically available by Pérez-Rodríguez et al. (2013) in the brnn package, consists of milk production records for 3 traits (fat yield, milk yield and protein yield) in Jersey cows. The incidence matrices with marker codes for additive and dominance effects (X and Z, respectively) were derived from 33,262 SNP on each of 297 individually genotyped cows. The necessary data are stored as R objects that can be accessed once the library snnR is loaded. The phenotypic information is stored in the data frame object pheno, and the additive genomic relationship matrix and the dominance genomic relationship matrix are stored in object G and D, respectively. The data set also includes a vector (partitions) that assigns observations to 10 disjoint sets; this vector could be used to perform a 10 fold cross validation. We can fit Eq. [3] using the entries of the G matrix as inputs, and fat yield, milk yield or protein yield as response variable using the function snnR. A NN model that includes additive and dominance effects jointly [5] was fitted using the G and D matrices of the Jersey data. The model can be fitted using the function snnR_extended. Details of the commands are in the manual of snnR.

Simulated Dataset

The second of the data sets consists of a simulated data including 10,000 F2 individuals from two founder lines with information 1,000 biallelic SNP loci, the loci

were evenly distributed over ten chromosomes of length 1M each. The mutation rate for both SNP marker and QTL was set to $2.5 * 10^{-5}$. The cumulative effect of the simulated QTL equaled the genetic value of the individuals, while the phenotypes of the individuals had been obtained as the sum of the individual's genetic value and random drawn from a normal distribution with mean zero and a variance set to produce different heritabilities: 0.1, 0.3, 0.5 and 0.7. The data also includes a file that assigned observations to 10 disjoint sets. We have also provided a simulated data including 10,000 F2 individuals from two founder lines with 20,000 biallelic SNP loci at the heritability 0.7 for testing the predictive performance of our snnR package for big samples and big input dimension. The data is available with a quick-start guide at: http://mgb.ouc.edu.cn/novegene/html/sdnn.php.

RESULTS

Evaluation of the Predictive Power

Jersey Dataset. We have compared the predictive performance of snnR, brnn(Pérez-Rodríguez et al., 2013) and RR-GBLUP (Meuwissen et al., 2001) with the two data sets. Table 1, Supplementary Table S1 and Supplementary Table S2 present the results of evaluation of the predictive power of the models including additive and dominance effects with a 10 fold cross-validation with the genomic relationship G matrix and D matrix in Jersey Dataset as inputs for milk yield, fat yield and protein yield, respectively. For different values of $\boldsymbol{\lambda}$, we trained a model on a training set and selected the value of $\boldsymbol{\lambda}$ that gives the smallest prediction error on a validation set as the optimal $\boldsymbol{\lambda}$. The correlation between observed and predicted values for each of the folds was obtained as the average of 10 runs in the case of snnR software to mitigate

the effect of the starting values for all parameters in the net. The last two rows of the tables show the average correlation and the average root mean square error (**RMSE)**.

For Jersey Dataset with only 297 samples, predictive ability of RR-GBLUP showed among the greatest in the case of predictions for the fat yield and the protein yield. However, for the remainder trait-the milk yield snnR with 3-hidden layers and 5 neurons in each hidden layer for each of the additive and dominance components produced the most accurate predictive ability about 2% greater than RR-GBLUP. RR-GBLUP was found to be more accurate than brnn. Wimmer et al. (2013) had compared the performance of four commonly used methods such as RR-GBLUP and BayesB (Meuwissen et al., 2001), LASSO (Tibshirani, 1996), and the elastic net (Zou and Hastie, 2005). They also found that good performance with RR-GBLUP due to long-range LD, medium heritabilities, and small sample sizes.

For the same NN methods, the predictions obtained by using snnR are better than those obtained by brnn with the same network architectures (Table 1 and Supplementary Table S1-S2). Our data show that the deep-structured NN have the advantage for predicting complex traits over shallow-structured single layer NN. Furthermore, on average, inclusion of the dominance component yielded slightly better predictions for milk yield (Table 1), even though milk yield was generally assumed to be an additive trait. This hypothesis is also supported by results of non-additive effects on milk yield in Jersey cows, which reported that individual

non-additive effects made a small contribution to the genetic variation of milk yield (Aliloo et al., 2015).

The great variability of the results was observed on the folds 7 and 9 with null or negative correlations for fat yield, milk yield and protein yield using all three methods(Table 1 and Supplementary Table S1-S2). A possible explanation for such variability could be overfitting caused by sampling in the small sample size of Jersey.

Simulated Dataset. Table 2 shows the results of evaluation of the predictive power of the models with additive effects for Simulated Dataset. RR-GBLUP achieved predictive abilities about 0.3417, 0.6068, 0.6791 and 0.8695 for the traits with heritabilities 0.1, 0.3, 0.5 and 0.7, respectively. For all traits, snnR and brnn reached predictive abilities between 7.91% and 19.37% and between 7.63% and 9.12% greater than RR-GBLUP, respectively. One possible explanation for the worse performance of RR-GBLUP could be the fact that RR-GBLUP was fitting with an inverse of G matrix (Meuwissen et al., 2001) while the dimension of the inverse of G matrix was too big (close to 10,000 in this example) to calculate correctly. For the trait with high heritability 0.7, snnR and brnn reached also the same good predictive abilities. With the heritability decreasing, the predictive ability of snnR obviously outperformed brnn and deep-structured snnR had the advantage for predicting complex traits over shallow-structured snnR. For an example, snnR has been able to achieve a predictive ability as high about 0.8872 when using the simulated dataset with 10,000 samples and 20,000 SNP genotype inputs.

Evaluation of the Computation Time

Both the NN methods were implemented in R and ran on a Linux machine with an Intel Core i5-3230M CPU processor @2.60GHz with 4 GB of RAM memory. Figure 1 presents the results of RMSE corresponding to the training computation time (in seconds) for brnn and snnR with different network architectures (2 to 15 neurons in 1-hidden layer) for genome-enabled predictions of milk traits in Jersey cows using the G matrix as input. We noticed that (1) RMSE of our snnR rapidly decreased as the network was trained with a relatively simple NN architecture (2 - 5 neurons in the hidden layer, Fig.1a-d) and even with complex NN architectures (over 5 neurons in the hidden layer, Fig.1e and f); (2) however, RMSE of brnn decreased very slowly with the increased number of neurons in the hidden layer of NN (Fig.1a-f). For each simulated dataset at the different heritabilities in Table 2, the brnn software took over 12 h to fit the model, while our snnR completed it within 30 min when both packages using the same simple NN architecture with 2 neurons in a single hidden layer.

Optimal Sparse Structure

Fig. 2a shows a nonlinear function. The predicted results of brnn with 1-hidden-layer architecture, snnR with 5-hidden-layer architecture, and snnR with 7-hidden-layer architecture (each layer have 5 neuros) are shown in Fig. 2b, Fig. 2c, and Fig. 2e, respectively. The optimal sparse structures of snnR with 5-hidden-layer architecture and 7-hidden-layer architecture, which were obtained by passing the parameters $wDNNs of snnR into the function plotnet of NeuralNetTools (Beck,

2015), are showed in Fig. 2d and Fig. 2f. The results demonstrated that deep-structured sparse NN have obvious greater accuracy in nonlinear function regression than shallow-structured single layer NN of brnn. Details of the commands are in the manual of snnR.

To build an appropriate and compact structure for a network, which involves making an optimal trade-off between the training data and model complexity, is a difficult task, due to the huge search space of possible models. Information criteria AIC (Akai Information Criteria; AIC= $-2/T\ln\left(\hat{\mathcal{L}}(\boldsymbol{X})\right)+2K/T$, where T denotes the number of observations and K denotes the number of parameters; Akaike, 1973) and BIC (Bayesian Information Criteria; BIC= $-2/T\ln\left(\hat{\mathcal{L}}(\boldsymbol{X})\right)+2K\ln(T)/T$; Schwarz, 1978) for neural networks have still been widely used to find an optimal trade-off between an unbiased approximation of the underlying model and the loss of accuracy caused by estimating an increasing number of parameters (Anders and Korn, 1999). Both AIC and BIC are effective in testing these models with the smaller their values indicating the higher effective structure of a network.

Figure 3 presents the combined results of AIC and BIC for all prediction scenarios tested. The single panels (a - h) show the dependency of the average AIC and BIC results of cross-validation runs on the optimal sparse NN models with different initialized fully-connected NN architecture(1 to 8 hidden layers and 2 to 10 neurons in each hidden layer) for milk yield as response and G used as input to our snnR. Consistency of the changes of AIC and BIC in responses to neuron numbers in all the

single panels (Fig.3) indicates that the sparse NN models are suitable to build an appropriate and compact structure for a network using AIC or BIC. The smaller values of AIC and BIC are found in the 3-hidden-layer and 4--hidden-layer (Fig. 3 c and d), implying that some deep-structured NN models have the advantage for predicting milk yield over shallow-structured single layer NN models. The big values of AIC or BIC are seen in the NN with over 9 neurons and over 7 of hidden layers (Fig. 3 g and h), suggesting that the NN will learn irrelevant details of the data due to over-fitting.

Therefore, our strategy for setting the number of hidden layer and the number of hidden neurons in each layer is as follows: Step 1, add an additional hidden neuron, a new optimal sparse structure is obtained and the new value either AIC or BIC is calculated. The hidden unit is accepted when criterion value AIC or BIC shows an improvement of their decreasing values. Step 2, when this is the case an enlarged network is estimated, new residuals and information criteria are computed. The procedure stops when an additional hidden neuron does not lead to further improvements based on the new information criteria AIC or BIC. Step 3, add a new hidden layer and repeat to do Step 1 and Step 2 until it does not lead to further improvements, then the appropriate and compact structure for a network has been eventually obtained.

Concluding Remarks

We have developed open source software snnR that allows fitting sparse NN in R and that works in Windows and UNIX-like environments. This software has potential usage in animal breeding: (1) It implements sparse NN to improve the prediction performance with input variables in high-dimensional space. Our method has resulted in lower prediction errors than the method of Bayesian regularized single layer neural networks for prediction of complex traits, especially with the bigger sample sizes. (2) It finds the optimal sparse network structure by minimizing the square error subject to a penalty on the L_1-norm of the parameters of a neural network to aggregate the high generalization capability and automatic parameter selection. (3) It raises computational efficiency due to using a smaller number of parameters in a sparse structure than in the brnn, especially dealing with input variables in high-dimensional space and the bigger sample sizes. (4) It extends the deep architecture feed-forward neural networks with many hidden layers to improve the modeling power to predict complex traits than when single layer architecture is used in NN. From Table 1 and Figure 2, we know deep-structured NN have better performance for predicting complex traits over shallow-structured single layer NN.

A main problem of snnR is probably its time consuming. Considering that computing the minimum-norm element of the subgradient is separable, our algorithm can be parallelized, hence to save computation time. Next, we will try to make our software much more effective by parallelizing over many machines with an asynchronous mode or over multiple GPUs (Lecun et al., 2015). For the optimization problem of sparse NN learning, it is obvious that better parameter initialization

techniques will lead to better prediction and computational efficiency. Thus the random initialization for the parameters of a neural network in this software has pretty room for improvement. The unsupervised pre-training technique which has attracted a wide attention in the parameter initialization may be a good alternative (Lecun et al., 2015). Moreover, we will implement the group spare regularization (Scardapane et al,2017) in our package snnR as well in future.

LITERATURE CITED

Akaike, H. 1973. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov, & F. Csaki (Eds.), Second International Symposium on Information Theory (pp. 267-281).Budapest: Akademiai Kiado.

Aliloo, H., J. E. Pryce, O. Gonzalezrecio, B. G., Cocks, and B. J. Hayes. 2015. Validation of markers with non-additive effects on milk yield and fertility in Holstein and Jersey cows. BMC Genet. 16:89-89. doi:10.1186/s12863-015-0241-9

Anders, U., and O. Korn. 1999. Model selection in neural networks. Neural Netw. 12:309-323. doi:10.1016/S0893-6080(98)00117-8

Beck, M. 2015. NeuralNetTools: Visualization and analysis tools for neural networks. Version 1.5.0. https://cran.rstudio.com/package=NeuralNetTools.

Bertsekas, D., A. Nedic, and A. Ozdaglar. 2003. Convex analysis and optimization. Athena Scientific, Belmont, MA.

Dennis, J., and J. More. 1997. Quasi-newton methods, motivation and theory. SIAM Rev Soc. Ind. Appl. Math. 19:46-89. doi:10.1137/1019005

Ehret, A., D. Hochstuhl, D. Gianola, and G. Thaller. 2015. Application of neural networks with back-propagation to genome-enabled prediction of complex traits in holstein-friesian and german fleckvieh cattle. Genet. Sel. Evol. 47:1-9. doi:10.1186/s12711-015-0097-5

Fletcher, R. 2001. On the barzilai-borwein method, Technical Report. University of Dundee. UK.

Gianola, D., and J. B. van Kaam. 2008. Reproducing kernel Hilbert spaces regression methods for genomic assisted prediction of quantitative traits. Genetics 178:2289-2303. doi:10.1534/genetics.107.084285

Gianola, D., H. Okut, K. A. Weigel, and G. J. M. Rosa. 2011. Predicting complex quantitative traits with bayesian neural networks: a case study with jersey cows and wheat. BMC Genet. 12: 87-100. doi:10.1186/1471-2156-12-87

Gill, P., W. Murray, and M. H. Wright. 1984. Practical optimization. Academic Press, London, UK.

Gripon, V., and C. Berrou. 2011. Sparse neural networks with large learning diversity. IEEE Trans. Neural Netw. Learn. Syst. 22:1087-96. doi:10.1109/TNN.2011.2146789

Lecun, Y., Y. Bengio, and G. Hinton. 2015. Deep learning. Nature 521:436-444. doi:10.1038/Nature14539

Krishnan, D., P. Lin, and A. Yip. 2007. A primal-dual active-set method for nonnegativity constrained total variation deblurring problems. IEEE Trans. Image Process. 16:2766-2777. doi:10.1109/TIP.2007.908079

Kurkova, V. 1992. Kolmogorov theorem and multilayer neural networks. Neural Netw. 5:501–506. doi:10.1016/0893-6080(92)90012-8

Meuwissen, T. H., B. J. Hayes, and M. E. Goddard. 2001. Prediction of total genetic value using genome-wide dense marker maps. Genetics 157: 1819-1829.

Nocedal, J. 1980. Updating quasi-newton matrices with limited storage. Math. Comput. 35:773-782. doi:10.1090/S0025-5718-1980-0572855-7

Okut, H., X. L. Wu, G. J. M. Rosa, S. Bauck, B. W. Woodward, and R. D . Schnabel. 2013. Predicting expected progeny difference for marbling score in angus cattle using artificial neural networks and bayesian regression models. Genet. Sel. Evol. 45:101-105. doi:10.1186/1297-9686-45-34

Pérez-Rodríguez, P., D. Gianola, K. A. Weigel, G. J. M. Rosa, and J. Crossa. 2013. Technical note: an R package for fitting bayesian regularized neural networks with applications in animal breeding. J. Anim. Sci. 91:3522-3531. doi:10.2527/jas.2012-6162

Scardapane, S., D., Comminiello, A., Hussain, and A. Uncini. 2017. Group Sparse Regularization for Deep Neural Networks. Neurocomputing 241:81-89. doi:10.1016/j.neucom.2017.02.029

Schwarz, G. 1978. Estimating the dimension of a model. Ann. Stat. 6:461-464. doi:10.1214/aos/1176344136

Thomaidis, N., and G. Dounias. 2011. On detecting the optimal structure of a neural network under strong statistical features in errors. J. Time Ser. Anal. 32: 204-222. doi:10.1111/j.1467-9892.2010.00693.x

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. J. Roy. Stat. Soc. B. 58: 267-288. doi:10.1111/j.1467-9868.2011.00771.x

VanRaden, P. M. 2008. Efficient methods to compute genomic predictions. J. Dairy Sci. 91:4414–4423. doi: 10.3168/jds.2007-0980

Wimmer, V., C. Lehermeier, T. Albrecht, H. J. Auinger, Y. Wang. 2013. Genetics 195:573–587. doi:10.1534/genetics.113.150078

Zou, H., and T. Hastie. 2005. Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. A Stat. Soc. 67:301–320. doi:10.1.1.124.4696

Table 1. Correlations between observed and predicted values of milk yield in a testing set of Jersey cows[1]

Fold	Additive				Additive + dominance		
	RR-GBLUP	brnn1-2-1	snnR1-2-1	snnR1-5-5-5-1	brnn1-2:2-1	snnR1-2:2-1	snnR1-5-5-5:5-5-5-1
1	0.5328	0.3455	0.4269	0.4459	0.2715	0.4189	0.4707
2	0.5894	0.5211	0.5862	0.6163	0.5546	0.6371	0.6486
3	0.4782	0.2553	0.4125	0.4808	0.4264	0.4063	0.5215
4	0.6965	0.5241	0.6594	0.6861	0.5051	0.6217	0.6763
5	0.2887	0.1118	0.2716	0.3115	0.0989	0.2984	0.2817
6	0.4880	0.3821	0.4336	0.4513	0.4211	0.4335	0.4878
7	-0.0668	-0.1365	-0.0622	-0.0464	-0.0756	-0.0132	-0.0115
8	0.7364	0.4533	0.6379	0.6886	0.5387	0.6578	0.7761
9	0.1415	0.1361	0.1322	0.1917	0.1974	0.1446	0.1897
10	0.5231	0.4334	0.5034	0.5451	0.4920	0.5328	0.5774
Avg. COR	0.4407	0.3026	0.4002	0.4372	0.3430	0.4138	0.4619

Avg.RMSE	50.73	68.79	54.28	52.17	67.93	53.84	48.89

[1]brnn 1-2-1 = Bayesian regularized neural network with 2 neurons in 1-hidden layer for the additive component; snnR 1-2-1 = Sparse neural network with 1-hidden layer and 2 neurons in the hidden layer for the additive component ; snnR 1-5-5-5-1 = Sparse neural network with 3-hidden layers and 5 neurons in each hidden layer for the additive component; brnn 1-2:2-1 =Bayesian regularized neural network with 2 neurons for each of the additive and dominance components; snnR 1-2:2-1 = Sparse neural network with 1-hidden layer and 2 neurons in the hidden layer for each of the additive and dominance components; snnR 1-5-5-5:5-5-5-1 = Sparse neural network with 3-hidden layers and 5 neurons in each hidden layer for each of the additive and dominance components; COR = correlation ; RMSE = root mean square error.

Table 2. Correlations between observed and predicted values for Simulated Dataset for traits with different heritabilities: 0.1, 0.3, 0.5 and 0.7[2]

Fold	Additive h^2=0.1				Additive h^2=0.5			
	RR-GBLUP	brnn1-2-1	snnR1-2-1	snnR1-5-5-5-1	RR-GBLUP	brnn1-2-1	snnR1-2-1	snnR1-5-5-5-1
1	0.3068	0.4109	0.4869	0.4993	0.6812	0.7614	0.8145	0.8477
2	0.3549	0.4052	0.4535	0.4606	0.6489	0.7774	0.8368	0.8668
3	0.3859	0.4277	0.4771	0.5166	0.7007	0.7754	0.8248	0.8716
4	0.3126	0.3736	0.4330	0.4985	0.6924	0.7601	0.8577	0.8896
5	0.3306	0.4680	0.4278	0.4931	0.6976	0.7780	0.8529	0.8831
6	0.3056	0.4729	0.4403	0.4976	0.7082	0.7542	0.8736	0.8969
7	0.3458	0.3864	0.4265	0.4501	0.6501	0.7687	0.8401	0.8531
8	0.3885	0.4131	0.4971	0.4432	0.6807	0.7719	0.8561	0.8724
9	0.3786	0.4401	0.4843	0.5516	0.6871	0.7696	0.8508	0.8774
10	0.3078	0.3961	0.4470	0.4905	0.6448	0.7863	0.8446	0.8691
Avg COR	0.3417	0.4194	0.4573	0.4912	0.6791	0.7703	0.8451	0.8728

Avg.RMSE	10.22	9.74	9.12	8.97	5.7125	3.29	1.71	1.52
	Additive h^2=0.3				Additive h^2=0.7			
Fold	RR-GBLUP	brnn1-2-1	snnR1-2-1	snnR1-5-5-5-1	RR-GBLUP	brnn1-2-1	snnR1-2-1	snnR1-5-5-5-1
1	0.6112	0.6745	0.7665	0.7651	0.8742	0.9484	0.9483	0.9449
2	0.6025	0.6516	0.7201	0.7371	0.8528	0.9388	0.9412	0.9516
3	0.6050	0.6762	0.7331	0.7611	0.8813	0.9499	0.9527	0.9445
4	0.6292	0.7105	0.7285	0.7969	0.8817	0.9428	0.9516	0.9531
5	0.6238	0.6853	0.7573	0.7221	0.8750	0.9454	0.9477	0.9465
6	0.6204	0.7199	0.7853	0.7583	0.8909	0.9439	0.9457	0.9460
7	0.5755	0.6772	0.7125	0.7997	0.8382	0.9454	0.9486	0.9376
8	0.6086	0.6504	0.7901	0.7446	0.8948	0.9525	0.9530	0.9642
9	0.6084	0.6664	0.7695	0.7751	0.8798	0.9502	0.9534	0.9585
10	0.5838	0.7197	0.7357	0.7419	0.8263	0.9427	0.9431	0.9428
Avg COR	0.6068	0.6831	0.7498	0.7601	0.8695	0.9460	0.9485	0.9489

Avg.RMSE	7.39	5.34	4.16	3.57	1.97	0.95	0.98	0.98

[2]brnn 1-2-1 = Bayesian regularized neural network with 2 neurons in 1-hidden layer for the additive component; snnR 1-2-1 = Sparse neural network with 1-hidden layer and 2 neurons in the hidden layer for the additive component; snnR 1-5-5-5-1 = Sparse neural network with 3-hidden layers and 5 neurons in each hidden layer for the additive component; COR = correlation; RMSE = root mean square error.

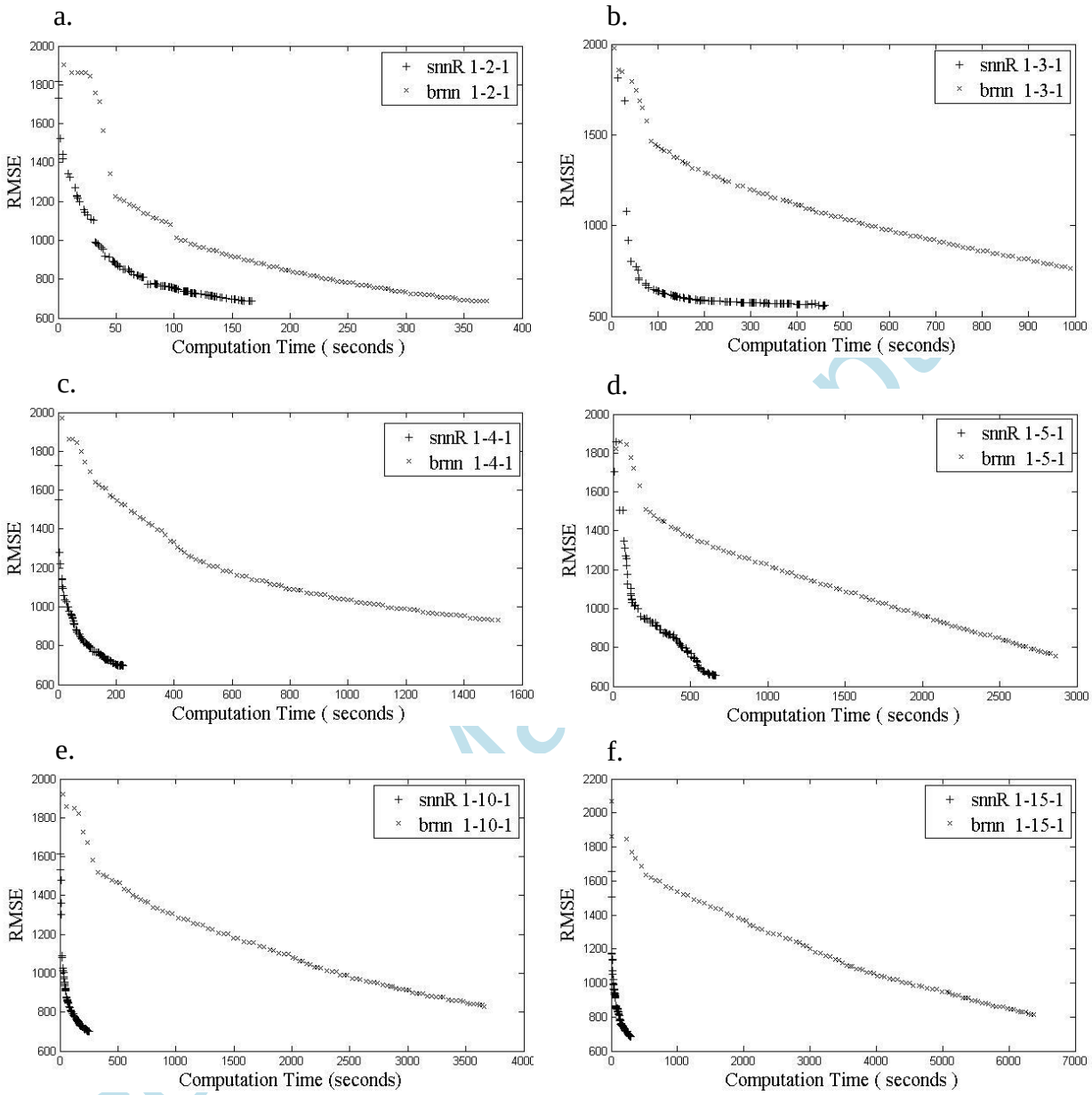

Figure 1 : Results of RMSE corresponding to the training computation time (in seconds) for brnn and snnR with different network architectures for genome-enabled predictions of milk traits in Jersey cows using the G matrix as input. (a)- (f) structures with brnn 1-n-1 = Bayesian regularized neural network with n from 2 to 15 neurons in 1-hidden layer; snnR 1-n-1 = Sparse neural network with 1-hidden layer and n from 2 to 15 neurons in the hidden layer. RMSE = root mean square error.

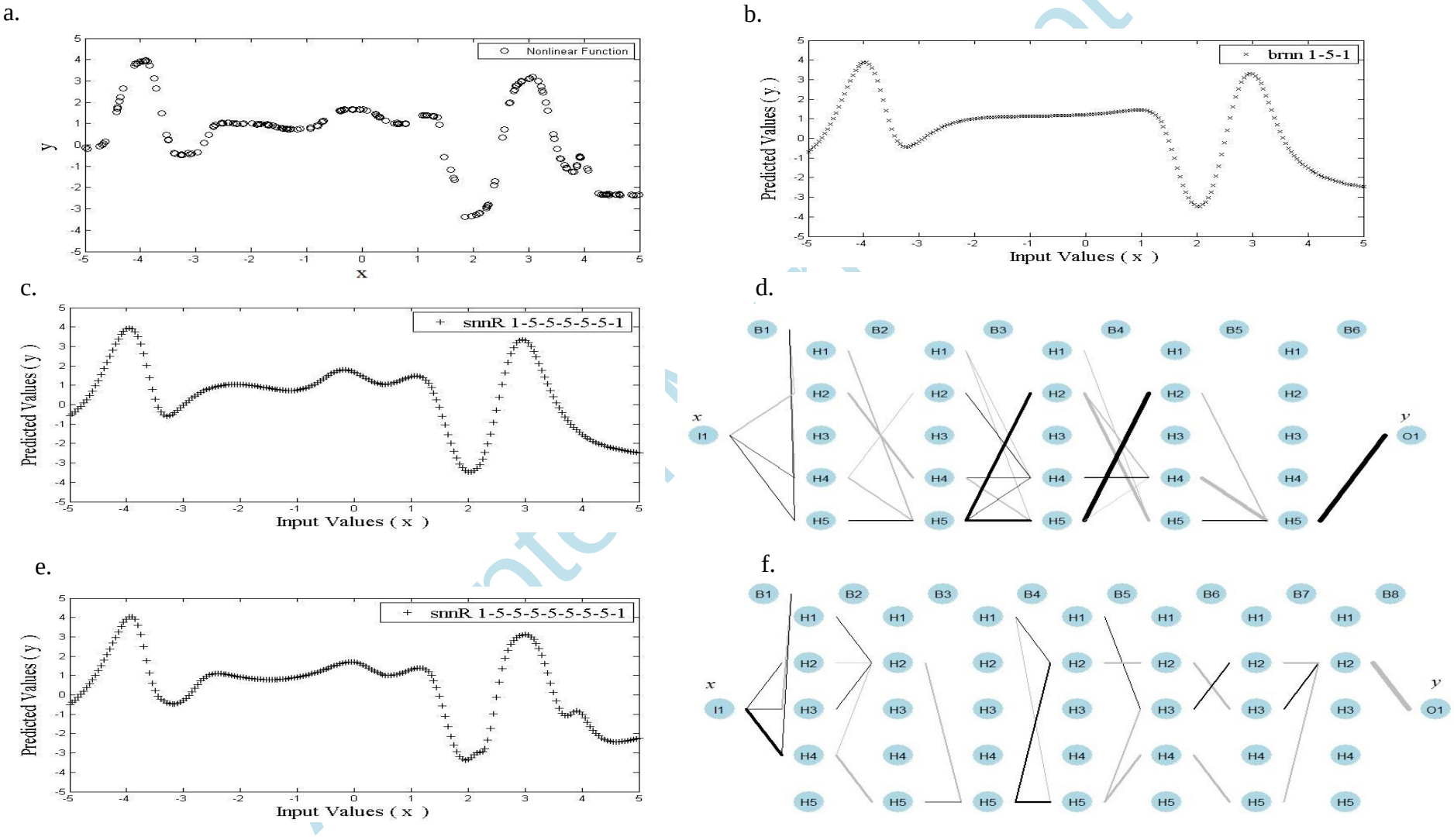

Figure 2 : Nonlinear regression. (a) a nonlinear function, (b) the predicted results of brnn with 1-hidden-layer architecture 5 neurons in the hidden layer., (c) the predicted results of snnR with 5-hidden-layer and 5 neurons in each hidden layer, (d) the optimal sparse structure of snnR with 5-hidden-layer, (e) the predicted results of snnR with 7-hidden-layer and 5 neurons in each hidden layer, (f) the optimal sparse structure of a snnR with 7-hidden-layer.

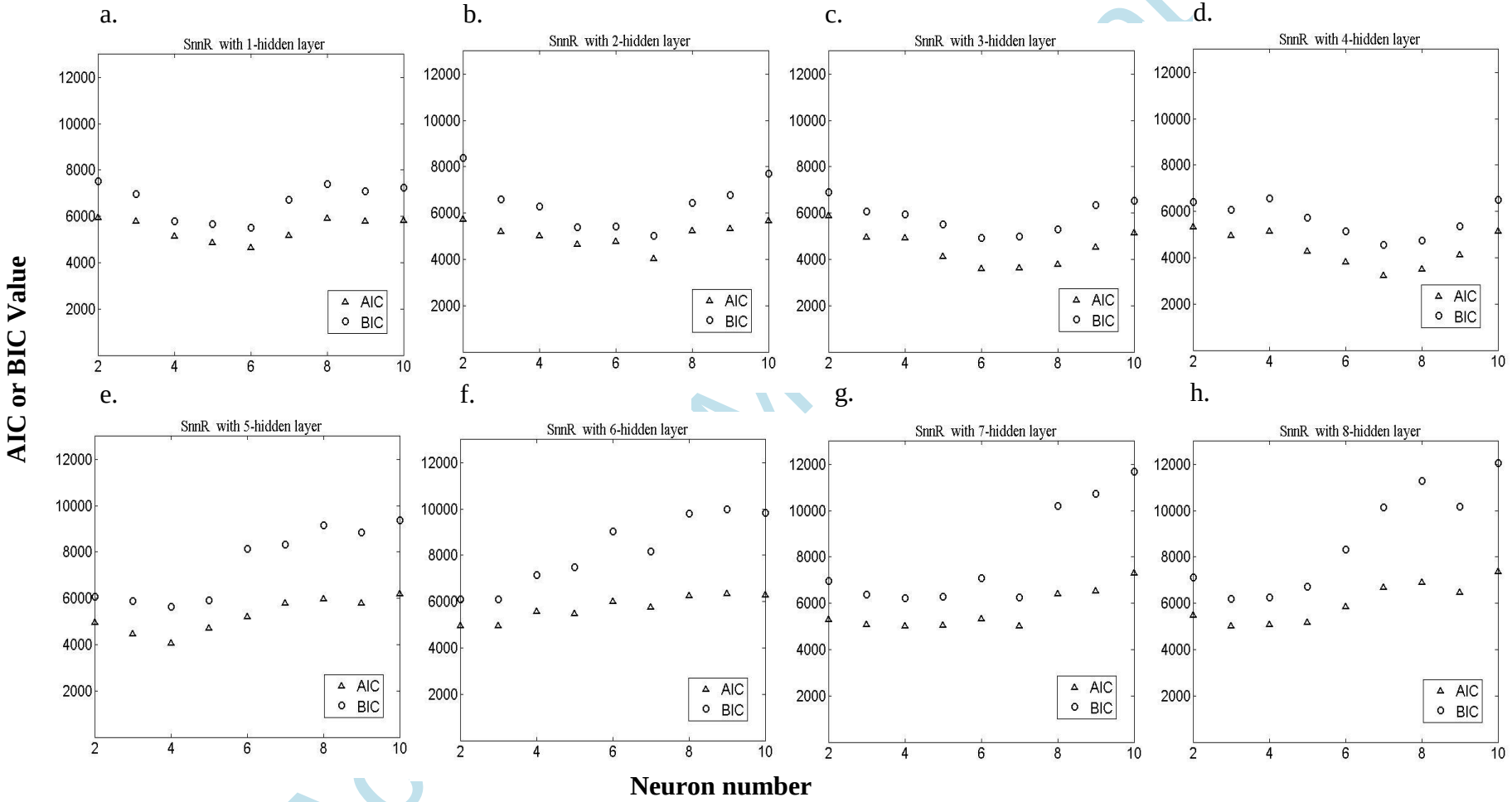

Figure 3：AIC and BIC evaluated using initialized NN with different hidden layers and different neurons in each layer for predicting values of milk yield in a testing set of Jersey cows. Panels (a- h) show the dependency of the average AIC and BIC results of cross-validation runs on the optimal sparse NN models with different initialized fully-connected NN architecture(1 to 8 hidden layers and 2 to 10 neurons in each hidden layer) for milk yield as response and G used as input to our snnR.